Timing-Driven Hierarchical Global Routing with Wire-Sizing and Buffer-Insertion for VLSI with Multi-Routing-Layer

Takahiro DEGUCHI† Tetsushi KOIDE‡ Shin'ichi WAKABAYASHI†

†Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1 chome, Higashi-Hiroshima, 739-8527 JAPAN
e-mail: wakaba@computer.org

‡VLSI Design and Education Center, The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656 JAPAN
e-mail: koide@vdec.u-tokyo.ac.jp

Abstract— In the high performance VLSI with multi-layer layout model, the complexity of the global routing problem becomes much high under timing constraints. This paper presents a hierarchical global routing method based on a multi-layer routing model for the high performance standard cell layout. In each hierarchical level, the routes of nets are determined by solving a linear programming problem considering wire-sizing and buffer-insertion under timing constraints. We have implemented the proposed method on a workstation and showed the effectiveness of the method from experimental results.

I Introduction

With recent advances of deep-submicron technologies, VLSI has been used in various kinds of electronics products. Nowadays, the number of transistors mounted on a state-of-the-art VLSI chip is about 10^8. In such a VLSI chip, interconnect design plays an important role in determining the chip performance [3], and in recent years, many interconnect optimization techniques, including interconnect wire sizing, driver sizing, buffer insertion and sizing, have been proposed and shown to be very effective for interconnection delay reductions [6, 13].

As the another concern of the routing process, the routing layers have become more than two. In the past, with one or two layers available for global routing, the job was simply to find the routing channels needed for each net. As the VLSI technologies evolved, the multi-layer (more than two layers) routing makes issues, such as via minimization and layer assignment, much more difficult. In the future, the number of layers of a chip will have more than six or eight [7].

In the high performance VLSI with the multi-layer layout model, the complexity of routing problem becomes much high under timing constraints. Although there have been many previous methods for the global routing problem in VLSI layout design, there have been few results on the problem with the multi-layer model. Eugene and Sechen have proposed a multi-layer chip-level global router [9], but their layout model is the macro-cell design style. Timing constraints have not also been considered in their method. So, in this paper, we propose a timing-driven hierarchical global routing method with wire-sizing and buffer-insertion for a multi-layer channel-less standard cell layout model. To estimate interconnection delay accurately, we adopt the delay estimation model based on Elmore's delay model [8].

In the proposed method, we treat a multi-layer model for very large scale circuits and route nets considering both the timing constraints and the routing congestion simultaneously. In the proposed routing method, first, the routing area is divided into sub-regions called blocks. Initially, layout area consists of 4×4 blocks. Each block at a certain level is divided into 4×4 recursively. Next, we route nets in each 4×4 region simultaneously by solving the linear programming problem, in which each possible route of a net is represented as a routing pattern meeting timing constraints considering wire-sizing and buffer-insertion. After 4×4 routing, we divide the routing area hierarchically and set terminals among sub-regions in the next hierarchy level. We repeat area routing and hierarchical decomposition until the predetermined level.

This paper is organized as follows. In Section II, we describe the layout model and graph model used in this paper. In Section III, we propose a timing-driven hierarchical global routing algorithm with wire-sizing and buffer-insertion. In Section IV, experimental results and the evaluation of the proposed algorithm are shown. Finally in Section V, we describe the conclusions and future researches.

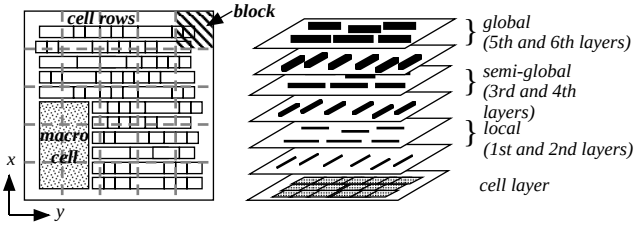
II Preliminaries

A. Layout Model

The layout model we assume in this paper is the cell based model which consists of row based standard-cells and some macro-cells. Moreover, a channel-less cell layout model can be used. We assume that the interconnections are done using six routing layers. However, it is possible to extend our routing method to the multi-layer layout model with more than six routing layers. The routing layer consists of a pair of vertical and horizontal wiring layers. We call ith pair of routing layer simply ith routing layer($i = 1, 2, 3$). The wire width and the wire space of ith routing layer are represented as w_i and s_i, respectively, and we define the wire pitch of ith routing layer as $g_i = w_i + s_i$. We assume the wire pitch satisfies $g_1 \leq g_2 \leq g_3$, that is, the wire pitch is larger than those of lower routing layers. So, we call each layer which has different routing pitch, *local*, *semi-global*, and *global routing layers*, respectively (Figure 1).

B. Graph Model

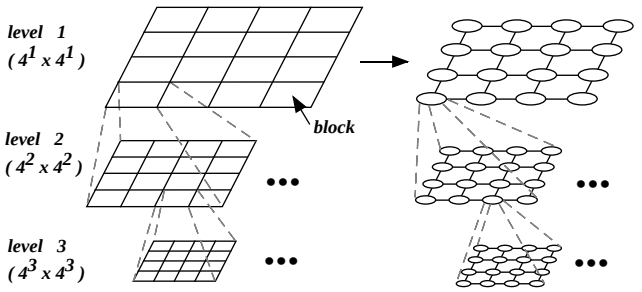
In our proposed method, we perform the global routing hierarchically. The entire layout area is divide into 4×4 regions. We call each region a *block*. The depth of hierarchy is called a



(a) chip layout *(b) multi-routing layers*

Fig. 1. Layout model.

level. Initially, on level 1, the layout area is divided into 4×4 blocks, and subsequently, on each level $l (1 \leq l \leq \mathcal{L})$, the whole chip area will be divided into $4^l \times 4^l$ blocks (Figure 2). On each level, a *border* is a line between two adjacent blocks.



(a) Block decompositon *(b) Hierarchical graph.*

Fig. 2. Hierarchical decomposition.

The routing area is represented by *hierarchical graph* $G_l = (V_l, E_l)$ at level l. A vertex $v_{ij_{(l)}} \in V_l$ represents the block with *x*-coordinate i and *y*-coordinate j. Each edge $e_{i,i+1_{(l)}} \in E_l$ connects adjacent vertices in the graph. The edges on the horizontal and vertical directions are represented as $e_{i\pm1,i_{(l)}}$ and $e_{i,i\pm1_{(l)}}$, respectively. Each vertex $v_{pq_{(l)}} \in V_l$ corresponds to a set of 4×4 vertices $V_{l+1}^{v_{pq_{(l)}}} = \{v_{ij_{(l+1)}} : 4p \leq i \leq 4p + 3, 4q \leq j \leq 4q + 3\}$ at level $l + 1$, and *decomposed graph* of $v_{pq_{(l)}}$, denoted as $DG_{l+1}^{v_{pq_{(l)}}} = (V_{l+1}^{v_{pq_{(l)}}}, E_{l+1}^{v_{pq_{(l)}}})$, is defined with the edges connecting adjacent vertices (Figure 3). Each edge

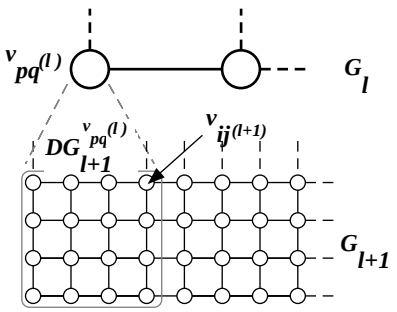


Fig. 3. Decomposed graph.

$e_{i,i+1_{(l)}} \in E_l$ has routing capacity for each routing layer. This provides constraints on the number of nets that can be routed through the edge. We define the routing capacity of $e_{i,i+1_{(l)}}$ on each routing layer as,

$$CAP_{i,i+1_{(l)}} = (cap^{(1)}_{i,i+1_{(l)}}, cap^{(2)}_{i,i+1_{(l)}}, cap^{(3)}_{i,i+1_{(l)}}).$$

If some routing area of the macro-cell region is prohibited to be used for routing, the routing capacity of the edges on the layer h may be set to zero : $cap^{(h)}_{i,i+1_{(l)}} = 0$.

The number of nets passing through the edge $e_{i,i+1_{(l)}}$ on each routing layer is represented as,

$$F_{i,i+1_{(l)}} = (f^{(1)}_{i,i+1_{(l)}}, f^{(2)}_{i,i+1_{(l)}}, f^{(3)}_{i,i+1_{(l)}}),$$

and the routing congestion of $e_{i,i+1_{(l)}}$ on each routing layer h $(1 \leq h \leq 3)$ as,

$$dns^{(h)}_{i,i+1_{(l)}} = cap^{(h)}_{i,i+1_{(l)}} - f^{(h)}_{i,i+1_{(l)}}.$$

In order to satisfy the routing capacity constraint, we must route all nets under the condition that $dns^{(h)}_{i,i+1_{(l)}} \geq 0$.

At each level l, a route of the net $n_j \in N'_l$ is represented as a Steiner-tree $T_l(n_j) = (V'_l, E'_l, L)$, where $V'_l \subset V_l$ consists of vertices corresponding to terminals of the net and Steiner points, $E'_l \subset E_l$ represents directed edges connecting vertices in V'_l, and L is a mapping of each edge to a layer to be used for routing, that is, the layer assignment of each edge. A global route of each net $n_j \in N$ corresponds to $T_{\mathcal{L}}(n_j)$ of each net at level $\mathcal{L}$.

III The Proposed Routing Algorithm

In this section, we will propose the timing-driven hierarchical global routing algorithm with wire-sizing and buffer-insertion. As mentioned above, we can treat a multi-layer routing model for interconnecting terminals. Since each routing layer has different wire width and wire spacing, we determine suitable wire width for each net so as to satisfy the timing and routing congestion constraints. We call this selection of wire width *wire sizing* in this paper.

The proposed routing algorithm is based on hierarchical routing [5, 15] and consists of three phases. In this approach, hierarchical decomposition of the problem into subproblems is performed so as to reduce the complexity of the overall global routing problem. The subproblems are solved, then the partial solutions are combined to produce a solution to the original global problem. In our routing algorithm, the top level corresponds to the entire layout, which is decomposed to 4×4 blocks in a top-down fashion. At each level of the hierarchy, we route a set of nets in each 4×4 regions simultaneously considering the interconnection delay and the routing congestion. We formulate each routing sub-problem as a 0-1 integer program. In the formulation of the 0-1 integer program, a variable is associated with a routing pattern of a net which satisfies the timing constraint with wire-sizing and buffer-insertion. After hierarchical decomposition reached to the lowest level $\mathcal{L}$, the routing congestion can be calculated and we rip-up some nets which violate the timing constraints and reroute them in the bottom up approach, taking into account the routing congestion.

Phase 1: At level l, find the routes of the nets in each 4×4 region by solving the 0-1 integer program.

Phase 2: Perform hierarchical decomposition and assign the virtual terminals at the border of each block.

Phase 3: Repeat Phases 1 and 2 until the lowest level $\mathcal{L}$. Then check the timing and routing congestion constraints, and reroute the nets violating constraints with a bottom up manner until no more improvement can be achieved.

In the following, we describe the details of each phase.

A. Phase 1 : 4×4 Area Routing

In the first phase, we route nets $N_l' \in N$ in 4×4 region at level l simultaneously so as to minimize the routing congestion by solving the 0-1 integer program. This routing was achieved on the 4×4 decomposition graph. Since it is very difficult and time-consuming to find all routes of nets in N_l' considering wire-sizing and buffer-insertion simultaneously, we select a subset of nets $N_l'' \subset N_l'$, formulate the 0-1 integer program for N_l'', solve the subproblem of finding routes for N_l'' simultaneously. We solve those subproblems in m_l times and obtain the whole routes of nets in N_l' at level l, where m_l satisfies $|N_l'| \leq m_l|N_l''|$. In the formulation of the 0-1 integer program, each 0-1 variable x_{ij} is associated with each possible routing pattern j of a net n_i, and we determine their patterns so that the routing congestion is minimized by selecting a particular set from many possible routing alternatives. We produce a set of routing patterns P_i of each net n_i which meets the timing constraints by wire-sizing and buffer-insertion so that the routing result given by the solution of the 0-1 integer program satisfies the timing constraints.

[4 × 4 Area Routing]

- **Step 1 :** Calculate the timing slack of N_l'.
- **Step 2 :** Select N_l'' from N_l' in decreasing order of the timing slack, and generate routing patterns for each net in N_l''.
- **Step 3 :** Find routes of nets in N_l'' by solving the 0-1 integer program.
- **Step 4 :** Repeat Steps 2 and 3 in m_l times.

1. Step 1 : Calculation of timing slack

Intrinsically, in the hierarchical routing approach, the routing of a net may not be completed in a 4×4 region and its terminals may include the virtual terminals (the virtual source and the virtual sink are explained in the next section), which are introduced in the hierarchical decomposition process. The interconnection delay, however, must be calculated based on the actual terminal positions to estimate the interconnection delay more exactly.

We estimate the interconnection delay from the given routing topology, in which the root of a subtree may be the virtual source and each wire is determined from the current routing solutions of each block(Figure 4). Moreover, if a block includes more than one terminals, we estimate the interconnection delay with making a subtree whose root is placed at the center of the coordinates of all the sinks.

The delay of each sink is calculated under the Elmore delay model [8], where each wire connecting two nodes has the Manhattan length. Since fringing capacitance has also been becoming a dominant factor of signal delay in sub-micron VLSI chips, we consider it explicitly in our delay model.

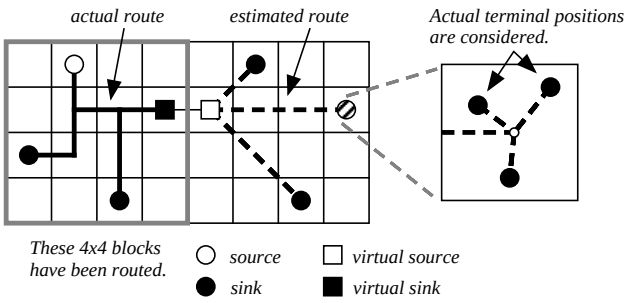


Fig. 4. Interconnection delay estimation.

The timing slack, denoted as $Sl(i,j)$, at sink i of net n_j, is presented as,

$$Sl(i\ j) = req(i\ j) - d(i\ j), \tag{1}$$

where $re\ q(i)$ and $d(i)$ are the required and actual delays of the sink i, respectively. If the value of the timing slack of sink i is larger than a threshold delay d_{th}, we call the sink i a *critical sink* of net n_j, and net n_j having critical sinks are called a *critical net*, where d_{th} is a constant given by the user.

The timing slack of net n_j is the minimum value of all the timing slack of sink i, and defined as,

$$Sl(n_j) = mi\ n\{Sl(i\ j)|\ \forall \text{sink } i \text{ of } n_j\} \tag{2}$$

So, the net which has a smaller slack value is more critical under given the timing constraints.

2. Routing Pattern Generation

Next, we will select a set of $|N_l''|$ nets from N_l' at level l in decreasing order of the timing slack of the net, and for each net we generate a set of routing patterns which are candidates of the net route. Each routing pattern p_{ij} of net n_j is different in its routing topology, in its wire width, or in the number of inserted buffers. For a set of routing patterns P_j of n_j, the input variables of 0-1 integer program are defined in the following step. The purpose in this step is to generate a set of different routing patterns of a net as much as possible, which satisfy the timing constraints considering wire-sizing and buffer-insertion. Since the number of possible routing patterns may be exponential in the size of the routing graph, we set the maximum number of routing patterns to P_{max} so as to reduce the computation time without degradation of the routing solution.

In generating the routing patterns of each net, we focus on both the interconnection delay and the routing congestion. For the critical nets, to reduce the delay using the wire-sizing and buffer-insertion, the minimum delay route will be generated. On the other hand, for the non-critical nets, routing patterns with many different topologies which satisfy the timing constraints are generated so that the chance of selection for reducing the routing congestion will become much high. Note that, because of the characteristics of the circuit data and the feature of the top-down hierarchical decomposition, more than half of all nets are two terminal nets. From these reasons, we

will propose two routing pattern generation algorithms, one for two terminal nets, and the other for nets having more than two terminals.

[Pattern Generation for Two-Terminal Nets]

Step 1: For each two terminal net n_k, repeat steps shown below.
Step 2: Generate routing pattern P_{jk} with the shortest path between terminals on local layer.
Step 3: If the net is critical,
then minimize the delay of P_{jk} by applying wire-sizing and buffer-insertion.
Step 4: Generate a routing pattern without using the edge in P_{jk} by weighted maze routing, where the weight of the edge is the routing congestion.
Step 5: If the generated routing pattern violates the timing constraints,
then minimize the delay by applying wire-sizing and buffer-insertion.

[Pattern Generation for Multi-Terminal Nets]

Step 1: For each multi-terminal net, repeat steps shown below.
Step 2: Generate a routing pattern as the minimum length tree T_{jk} on local layer(Fig.5(a)).
Step 3: If the net is critical, then make the minimum length tree without a critical sink, connect source and a critical sink with the shortest path(Fig.5(b)).
Step 4: Select a pair of sinks with which the sum of their capacitances is minimized, and make a subtree by weighted maze routing(Fig.5(c)). Repeat this procedure until the tree is constructed.
Step 5: If a generated routing pattern violates the timing constraints, then minimize the delay by applying wire-sizing and buffer-insertion(Fig.5(d)).

For the multi-terminal net, it is difficult to find the routing pattern as the Steiner tree satisfying the timing constraints. In generating a new generated routing pattern, we will check the timing constraints and if it violates the timing constraint, then it is abandoned. After applying wire-sizing and buffer-insertion to reduce the delay, the timing constrains may be satisfied.

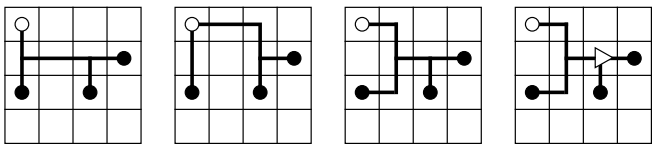


Fig. 5. Example of routing patterns of a multi-terminal net.

3. *Step 3 : Routing by 0-1 integer program*

In each 4×4 area routing, all the selected nets N_l'' are considered simultaneously instead of being routed in a particular net ordering. Candidate routes p_{ij} which are generated in the pattern generation step are identified and represented as 0-1 variables x_{ij}. For the nets in N_l'', each actual route is determined by solving the 0-1 integer program. The formulation of the 0-1 integer program is as follows:

[Routing by 0-1 integer program]
maximize Y
subject to

$$\sum_{1\le j\le |P_i|} x_{ij} = 1 \qquad i = 1,\cdots,\lambda$$

$$X(u,v,k) + \sum_{p_{ij}\in A(u,v,k)} x_{ij} = \delta\cdot cap^{(k)}_{uv_{(l)}} \qquad \forall u,v,k$$

$$\sum_{\forall n_i\in N''}\sum_{p_{ij}\in P_i} B(i,j,p,q) \le S(v_{(pq)_{(l)}}) \;\; \forall v_{pq_{(l)}} \in V_l'$$

$$Y \le X(i,j,k) \quad \forall i,j,k$$

$$x_{ij} \in \{0,1\}$$

where Y is the minimum routing slack, P_i is the set of routing patterns of net n_i, $X(u,v,k)$ means routing slack of $e^{(k)}_{ij_l}$, $B(i,j,p,q)$ stands for the number of buffers used to p_{ij} in v_{pq_l}, $S(v_{pq_{(l)}})$ represents the number of allowable buffers in v_{pq_l}, and $A(u,v,k)$ is a set of the routing patterns which pass through the edge $e_{uv_{(l)}}$ on kth layer.

Since we will solve the subproblem of routing for N', the routing capacity is given as $\delta\cdot cap^{(k)}_{uv_{(l)}} (0\le\delta\le 1)$. Although 0-1 integer program produces an optimal solution if it is solved, the computation time would be very large. So, we consider the relaxation of this formulation, in which integer constraints are replaced with $0 \le x_{ij} \le 1$. An obtained optimal solution by solving this linear program is a fractional solution. We use a randomized rounding algorithm [11, 12] to obtain an integer solution from a fractional solution.

B. *Phase 2 : Hierarchical Decomposition*

In this phase, the level of hierarchy is down to the next lower level, and we decompose each block at the current level to 4×4 blocks at the next level. At that time, the virtual terminals are placed on the boarder of blocks to map the obtained routes of nets to the next level (Figure 6).

In this section, we call a pair of blocks to which the virtual terminals are assigned a *slot*, and denote it as $u_i (i = 1,2,3,4)$. As shown in Figure 6, a slot consists of a pair of the adjacent blocks on the boarder of 4×4 regions at level $l+1$, where the boarder means the rectangle line of a block in level l.

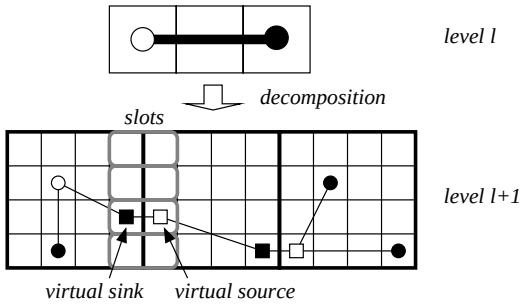


Fig. 6. Slots and virtual terminals.

When hierarchical decomposition is done, the route of a net split by decomposition must be reconnected by a pair of virtual

terminals, which are placed on a slot. We have to assign a net to one of four possible slot positions with which the net is passing through the boarder. Our approach uses linear assignment [4, 10] to this slot assignment problem. For each net n_j and any candidate slot u_i, we estimate the cost $cost(n_j, u_i)$ resulting from that assignment. This cost function is classified into the following three cases:

Case 1: $cost(n_j, u_i) = \alpha \cdot length(term, s)$. The virtual terminal is connected directly to the terminals of a cell.

Case 2: $cost(n_j, u_i) = \beta$. The virtual terminal is connected to a virtual terminal.

Case 3: $cost(n_j, u_i) = \gamma \cdot length(cut, s)$. The virtual terminal connects to the end of a segment.

where *length(term,s)* is the length between the terminal of a cell and a slot, *length(cut,s)* is the length between a boarder of a segment passing through and a slot, and α, β, γ are predetermined constants.

We estimate the cost from both the right and left sides $cost_l, cost_r$, and the total cost function is $cost(n_j, u_i) = cost_l(n_j, u_i) + cost_r(n_j, u_i)$. We formulate the linear assignment problem using this cost function, and solve it by an algorithm [4] to determine the positions of virtual terminals. Assignment of virtual terminals on vertical direction is done in a similar manner.

C. Phase3: Bottom Up Rerouting

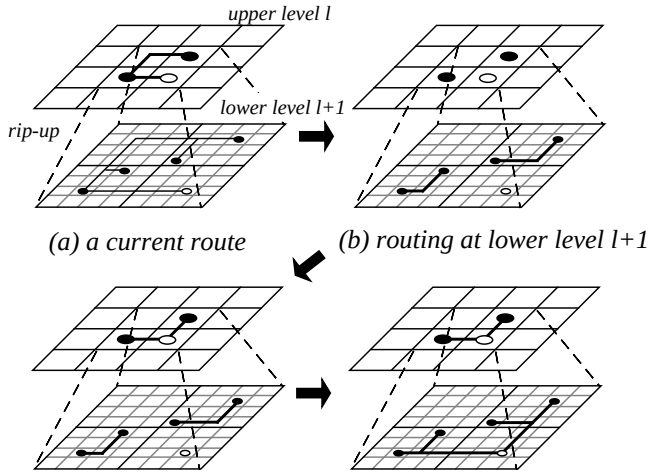


Fig. 7. Bottom up rerouting.

After repeating Phases 1 and 2 until the lowest level $\mathcal{L}$, although all nets are routed, some nets might cause the timing violation due to the discrepancy of estimation of the interconnection delay and the net based timing constraints(Fig.7). In Phase 3, we rip-up these nets and reroute them so as to satisfy the given timing constraints. When rerouting the nets, we use a bottom up routing approach considering the routing congestion in which almost nets have already been routed. In bottom up approach, the routes of nets are determined from the lowest level $\mathcal{L}$.

After ripping up nets which violate the timing constraints, first, we will route them inside of a block at level l+1 (Fig.7(b)). This is done on the decomposition graph of a block, that is, on hierarchical graph at level $l+1$ where weighted maze routing is performed to find a route with the minimum routing congestion. Next we must find a route at level l (Fig.7(c)), at the upper level, and we find the minimum delay route. Finally, this obtained route is rerouted at level $l+1$ as the same procedure in Phase 2.

IV Experimental Results

The proposed global routing algorithm has been implemented in C language, and run on a Ultra Comp Station model 170 with SunOS5.5. We have experimented it to the ISCAS/MCNC benchmarks shown in Table I. For these benchmarks, logic synthesis and technology mapping were performed by SIS1.2 [14] and the placements were generated by TimberWolf7.0 [2].

In this table, "#constraints" is the number of timing constraints. "avq_large×4" was generated by combining four "avq_large" circuits as an example of the large scale circuit which has more than 100,000 cells and nets. Table II shows the delay parameters used in the experiments.

TABLE I
Circuit data

Data	#cells	#nets	#I/O	#constraints
s38417	7572	7600	134	2619
s38584	8622	9002	342	2729
s35932	11838	11873	355	2823
avq_large	25114	25384	64	3422
avq_large×4	100456	101536	176	13688

TABLE II
The delay parameters ($0.18\mu m$ CMOS model).

Parameters	value
Min Gate Resistance (Ω)	280
Wire Resistance α ($\Omega/\mu m$)	0.076
Wire Capacitance β ($fF/\mu m$)	0.044
Fringing Capacitance cf ($fF/\mu m$)	0.055
Buffer Gate Delay ($nsec$)	0.125
Buffer Output Resistance(Ω)	210
Buffer Input Capacitance(fF)	1.0
Via delay ($nsec$)	0.022

In the following, we describe the parameters used in the experiments. Since the hierarchy level determines the size of the global routing gird, we will decompose the chip area until the size of a block has become the standard cell size (level 5). The 0-1 integer program in Phase 2 runs under the case that the number of nets ($|N_l''|$) is 500, and the maximum number of patterns for a net (P_{max}) is 30. We solve the relaxed linear program by using the linear programming package LOQO [1]. In Tables III and IV, we show the experimental results. In these tables, "wire_length" is the total wire length (μm). "wire_ratio"

is the completion ratio of routing, i.e., the ratio of the number of routed nets satisfying timing constraints to the total number of nets (%). "delay" is the maximum delay in the circuit ($nsec$). "#buffer" is the number of buffers which were inserted, and "CPU" is CPU time(sec).

First, Table III shows the results after Phase 2 at the lowest hierarchy level, that is, these results are obtained by only top-down hierarchical routing with wire-sizing and buffer-insertion without rerouting. From this table, the almost nets were routed in hierarchical routing with satisfying the timing constraints so that the rerouting process is needed to run in a short computation time. The complete routing results after rip-up and rerouting satisfy the timing constraints for all benchmark data. The computation time is reasonable for large scale circuits which have more than 100,000 nets.

TABLE III
ROUTING RESULTS AFTER PHASE2.

data	wire_length	wire_ratio	#buffer	CPU
s38417	1331421	98.5	43	513
s38584	2357728	96.4	89	3109
s38932	3281321	97.2	128	4918
avq_large	8932138	97.3	381	8830
avq_large×4	17898376	96.3	837	43296

We have also compared the routing results with the ones obtained by the method without wire-sizing and buffer-insertion. In the case of no wire-sizing and no buffer-insertion, the wire width of each layer is the same as that of local layer, that is, all layers have the minimum wire width. The routing patterns containing inserted buffers were not generated. Table IV shows the routing results after hierarchical routing(Phase 3). From Tables III and IV, the proposed method effectively produces the routing results satisfying the timing constraints. We also note that, without wire-sizing and buffer insertion, no feasible routing solutions can be obtained.

TABLE IV
ROUTING RESULTS WITHOUT WIRE-SIZING AND BUFFER-INSERTION.

data	wire_ratio	delay	CPU
s38417	63.5	5.42	328
s38584	72.4	8.32	2232
s38932	86.2	8.92	3874
avq_large	62.9	9.93	6328
avq_large×4	72.7	9.21	38789

V CONCLUSIONS

In this paper, we proposed a timing-driven hierarchical global routing algorithm with wire-sizing and buffer-insertion. The proposed algorithm assumes the multi-layer layout model which has different wire width on each routing layer. From the experimental results, the proposed routing algorithm successfully produces all routes of nets with satisfying the timing constraints.

For future research, we will improve the routing pattern generating algorithm, because quality of routing patterns determines the overall performance of the proposed algorithm. Extension of the proposed algorithm to the one in which crosstalk is taken into account is another important subject.

ACKNOWLEDGEMENTS

This research was supported in part by Grant-in-Aid for Encouragement of Young Scientists No. 10750248 from the Ministry of Education, Science, Sports and Culture, Japan.

REFERENCES

[1] "LOQO A system for solving linear and/or quadratic programing problems," Princeton University (1992).

[2] "TimberWolf 7.0 Macro/Standard Cell Floorplanning, Placement and Routing Package," Yale University (1996).

[3] H. B. Bakoglu: "Circuits, Interconnects, and Packaging for VLSI," Addison-Wesley (1990).

[4] R. E. Burkhard and U. Derigs: "Assignment and Matching Problems: Solution Methods with Fortran Programs," Springer-Verlag (1980).

[5] M. Burstein and R. Pelavin: "Hierarchical wire routing," IEEE Trans.CAD, vol.CAD-2, No.4, pp. 223–234 (1983).

[6] C.-P. Chen, Y.-P. Chen and D. F. Wong: "Optimal wire-sizing formula under the Elmore delay model," Proc. of the 33rd ACM/IEEE Design Automation Conference, pp. 487–490 (1996).

[7] J. A. Davis and J. D. Meindl: "Is interconnect the weak link? : Estimating wiring requirements of future-generation devices to meet NTRS needs," IEEE Circuits and Devices, 14, 2, pp. 30–36 (1998).

[8] W. C. Elmore: "The transient response of damped linear networks with particular regard to wideband amplifiers," J. Appl. Phys., Vol.19, pp. 55–63 (1948).

[9] L. C. Eugene and C. Sechen: "A multi-layer chip-level global router," Proc. 5th ACM/SIGDA Physical Design Workshop, pp. 218–225 (1996).

[10] G.Meixner and U.Lauther: "A new global router based on flow model and linear assignment," Proc. of International Conference on Computer-Aided Design, pp. 44–47 (1990).

[11] R. C. C. IV and C.-K. Cheng: "A global rouer using an efficient approximate multicommodity multiterminal flow algorithm," Proc.of the ACM/IEEE 28th Design Automation Conference, pp. 316–321 (1991).

[12] E. L. Lawler: "Combinatorial Optimization: Networks and Matroids," Holt, Rinehart and Winston (1976).

[13] J. Lillis, C.-K. Cheng, T.-T. Y. Lin and C.-Y. Ho: "New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing," Proc. of the 33rd ACM/IEEE Design Automation Conference, pp. 395–400 (1996).

[14] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli: "SIS:A system for sequential circuit synthesis," Tech. Electronics Research Laboratory (1992).

[15] N. A. Sherwani: "Algorithms for VLSI Physical Design Automation," Kluwer Academic Publishers, Sects. 4-5, pp. 123–203 (1993).